

Wilhelm-Ostwald-Schule, Gymnasium der Stadt Leipzig

Dokumentation zur Besonderen Lernleistung

im Fachbereich: Informatik

Thema: Webentwicklung zur Analyse der
Datenübertragung von Satelliten zu
Bodenstationen

vorgelegt von: Hannes Diener

Schuljahr: 2019/2020

Externe Betreuerin: Frau Diana Peters
Institut für Datenwissenschaften
Deutsches Zentrum für Luft- und
Raumfahrt

Interner Betreuer: Herr Rai-Ming Knospe

Leipzig, 11.01.2021

Kurzfassung

Bei der Planung von Satellitenmissionen ist die Auswahl geeigneter Bodenstationen zur Datenübertragung ein wichtiger Erfolgs- und Kostenfaktor.

Das Ziel dieser Besonderen Lernleistung war die Entwicklung einer nutzerfreundlichen Webanwendung zur Planung von wissenschaftlichen Satellitenmissionen. Wesentlicher Teil der Besonderen Lernleistung war die Ermittlung der optimalen Abfolge von Kontakten zwischen Satellit und Bodenstationen zum Erreichen der maximal übertragbaren Datenmenge.

Im Rahmen dieser Arbeit wurden die geografischen Daten und Antennenparameter vorhandener Bodenstationen in einer Datenbank zusammengetragen und für die Webapplikation aufbereitet.

Die Bahnparameter des Satelliten zur Berechnung des Satellitenorbits werden zusammen mit den Antennenparametern des Satelliten vom Anwender vorgegeben.

Für die Umsetzung der Webanwendung wurden JavaScript, HTML, CSS und das in der Programmiersprache Python geschriebene Webframework Django verwendet.

Inhaltsverzeichnis

1	EINLEITUNG	4
2	GRUNDLAGEN	7
2.1	Datenübertragung	7
2.1.1	Annahmen	7
2.1.2	Berechnungsalgorithmus	7
2.2	Benötigte Daten	9
2.2.1	Antennenparameter des Senders	9
2.2.2	Antennenparameter des Empfängers	9
2.2.3	Abstand von Bodenstation und Satellit	9
2.2.4	Signal-Rausch-Verhältnis	10
2.3	Verwendete Technologien	10
2.3.1	Django	10
2.3.2	Skyfield	11
2.3.3	HTML	11
2.3.4	CSS	11
2.3.5	Java Script	11
2.3.6	Cesium	11
3	DIE DATENBANK	12
3.1	Entitäten und Beziehungen	12

3.2	Attribute.....	12
3.2.1	Link.....	12
3.2.2	Antenne	12
3.2.3	Bodenstation und Betreiber	13
3.3	Entity-Relationship-Modell	13
3.4	Implementierung in Django	14
3.5	Datenbankabfragen.....	16
3.6	Population mit Daten.....	16
3.7	Zugriff auf die Datenbank durch den Nutzer	17
4	DIE ANWENDUNG	17
4.1	Grundlegende Funktionalität	17
4.1.1	Analysemodi.....	18
4.2	Django Template-System.....	18
4.3	Eingabe.....	20
4.3.1	Eingabewerte.....	20
4.3.2	Umsetzung	24
4.4	Analyse Datenübertragung	31
4.4.1	Ermittlung der Kontakte	31
4.4.2	Berechnung der Datenmengen.....	34
4.4.3	Ermittlung der optimalen Kontaktfolge.....	35

4.5	Ausgabe	38
4.5.1	Ausgabewerte	38
4.5.2	Wechsel zwischen Eingabe und Ausgabe	38
4.5.3	Ausgabe in Cesium	40
4.6	Laufzeit	43
5	FAZIT UND AUSBLICK	47
6	ABKÜRZUNGSVERZEICHNIS	48
7	FACHWORTERKLÄRUNGEN	49
7.1	Astronomie und Satellitenmissionen	49
7.2	Physikalische Größen	49
8	LITERATURVERZEICHNIS	51
9	ABBILDUNGS- UND QUELLTEXTVERZEICHNIS	52
9.1	Abbildungsverzeichnis	52
9.2	Quelltextverzeichnis	54
	DANKSAGUNG	56
	SELBSTSTÄNDIGKEITSERKLÄRUNG	57

1 Einleitung

Der erste erfolgreiche Start eines Satelliten am 4. Oktober 1957 durch die Sowjetunion (NASA, 2020) war der Startschuss für den Wettlauf ins All zwischen den Vereinigten Staaten von Amerika und der Union der Sozialistischen Sowjetrepubliken.

Nachdem die USA am 20. Juli 1969 mit der Mondlandung von Apollo 11 das Space Race für sich entscheiden konnten, veränderte sich der Fokus von Raumfahrtmissionen zunehmend hin zu wissenschaftlichen Zielen. Beispielsweise ermöglichen die Erdbeobachtungssatelliten der Mission Sentinel-6 des Deutschen Zentrums für Luft- und Raumfahrttechnik die Überwachung des Anstieges des Meeresspiegels und Erfassung von Meeresströmungen. Die Mission soll durch ihren Beitrag zur Klimaforschung das Leben auf der Erde langfristig verbessern.

Für die Durchführung einer solchen Satellitenmission werden ein Missionskontrollzentrum und ein Netzwerk an Bodenstationen benötigt (Ohndorf, 2015). Die Hauptaufgabe des Missionskontrollzentrums ist die Überwachung des Satellitenstarts und die Planung von Kontakten mit dem Satelliten. Eine Bodenstation umfasst meist mehrere Antennen, welche in verschiedenen Frequenzbereichen zur Übertragung von Funksignalen und Messdaten genutzt werden können. Dabei werden die Übertragungswege zwischen Antennen und Satelliten in Uplink und Downlink unterteilt. Als Uplink wird die Datenübertragungsrichtung von der Bodenstation zum Satelliten bezeichnet. Uplink beinhaltet vor allem Befehle an den Raumflugkörper. Der Datenfluss vom Satelliten zur Bodenstation heißt Downlink. Dieser umfasst die als housekeeping data bezeichnete Daten über die Systeme des Satelliten, primär aber die durch die Mission anfallenden Messdaten - die Nutzlastdaten.

Schon während der Planung ist eine Abstimmung auf die voraussichtlich zu übertragenen Datenmengen wichtig. Auf Seite des Satelliten müssen geeignete Systeme gewählt werden, um die erwarteten Datenübertragungsraten zu gewährleisten. Dabei muss ein Kompromiss

zwischen den erreichbaren Übertragungsraten und anderen Aspekten wie Masse, Platzbedarf und Energieverbrauch des Satelliten getroffen werden. Gegebenenfalls kann auch der für die Mission gewählte Orbit durch die Verfügbarkeit von Bodenstationen beeinflusst werden.

Diese Besondere Lernleistung beschäftigt sich mit der Entwicklung einer Webanwendung zur Auswahl geeigneter Bodenstationen für eine Satellitenmission. Dazu wurde als Grundlage eine Datenbank mit Bodenstationen erstellt. Auf die Eingabe von Missions- und Satellitendaten werden die im Downlink erreichbaren Datenübertragungsmengen ausgegeben. Anhand dessen kann eine Auswahl von Antennen, Bodenstationen oder Betreibern von Bodenstationsnetzwerken getroffen werden, welche zur Übertragung der Daten für eine Satellitenmission genutzt werden können.

Die als Ergebnis dieser Arbeit erstellte Webanwendung soll die Missionsplanung vereinfachen und verkürzen, indem das effektive Überprüfen von Ideen auf ihre theoretische Realisierbarkeit ermöglicht wird. Das ist insbesondere in der Konzeptionsphase von großer Bedeutung, wo viele verschiedene Entwürfe für Satellitenmissionen konkurrieren. Aufwendige manuelle Berechnungen der Datenübertragungsmengen, sogenannte Link Budget Analysen, sollen mit dem Einsatz der Webanwendung so weit wie möglich entfallen. Der Kostenfaktor selbst wird in dieser Arbeit nicht beachtet und die entwickelte Anwendung soll nicht die finale Prüfung der Machbarkeit der Mission ersetzen.

Die angestrebte universelle Verwendbarkeit sowie Einfachheit und Zeitersparnis beim Nutzen dieser Anwendung impliziert die Notwendigkeit einer nutzerfreundlichen Benutzeroberfläche.

Diese Besondere Lernleistung wurde durch das Institut für Datenwissenschaften des Deutschen Zentrums für Luft- und Raumfahrt (DLR) als externer Partner betreut.

Die im fachpraktischen Anteil erstellte Anwendung befindet sich auf der beigelegten CD. Das Starten der Anwendung wird in der Datei README.txt erklärt.

2 Grundlagen

2.1 Datenübertragung

2.1.1 Annahmen

Die Datenübertragung zwischen Satellit und Bodenstation erfolgt über Funk mit elektromagnetischen Wellen (Huber, 2015), und deren Simulation ist ein Kernbestandteil der zu entwickelnden Anwendung. Zur Vereinfachung wurde für die Entwicklung der Anwendung angenommen, dass ein Satellit genau eine Antenne besitzt und nur mit einer Antenne einer Bodenstation gleichzeitig kommunizieren kann.

Die für die Datenübertragung nutzbaren Frequenzen sind von großer Bedeutung. Sie werden durch die sogenannten Frequenzbänder begrenzt, welche durch die Internationale Fernmeldeunion (ITU) festgelegt werden. Eine Antenne, welche in mehreren Frequenzbändern genutzt werden kann, besitzt für jedes Frequenzband einen sogenannten Link. Ein Link beschreibt den Frequenzbereich, welchen die Antenne im Frequenzband abdeckt. Links können nach der Datenübertragungsrichtung in Up- und Downlinks eingeteilt werden. Bei wissenschaftlichen Satellitenmissionen machen die Sensordaten im Downlink den Großteil der übertragenen Datenmengen aus. Die Datenmenge der Befehle an den Satelliten im Uplink ist vergleichsweise gering. Deshalb wird in dieser Anwendung aus Gründen der Einfachheit ausschließlich die Datenübertragung von Satellit zur Bodenstation, also der Downlink, betrachtet.

2.1.2 Berechnungsalgorithmus

Die genaue Berechnung der theoretisch übertragbaren Datenmengen mitsamt Störeinflüssen, Modulation, Polarisation, Dopplereffekt und Codierung ist außerordentlich komplex und übersteigt den Rahmen dieser Arbeit. Es wurde deswegen die Möglichkeit geschaffen, in der Funktion `orbitcalc.general_utility.max_data_rate_calculation()` einfach einen exakteren Algorithmus zur Berechnung der Datenübertragung zu implementieren. Aufgrund der Wichtigkeit der Datenmengenberechnung für

die zu entwickelnde Anwendung, wurde daher ein selbst entwickelter Berechnungsalgorithmus zum Testen und zur Veranschaulichung der Ergebnisse genutzt. Dieser Algorithmus wurde vereinfacht aus einem Tool zur Link Budget Analyse des German Space Operations Center (GSOC) übernommen.

In diesem Algorithmus zur Berechnung der maximal möglichen Datenübertragungsrate C_{max} kommen folgende physikalischen Größen vor:

- minimale Frequenz des Empfängers $f_{min, \text{Empfänger}}$
- maximale Frequenz des Empfängers $f_{max, \text{Empfänger}}$
- minimale Frequenz des Senders $f_{min, \text{Sender}}$
- maximale Frequenz des Senders $f_{max, \text{Sender}}$
- Bandbreite B
- Freiraumdämpfung $FSPL$
- Vakuumlichtgeschwindigkeit c
- Abstand von Sender und Empfänger d
- EIRP des Senders
- G/T des Empfängers
- Boltzmann-Konstante k_B
- mindestens benötigte Eb/N0 und Margin

Berechnungsalgorithmus in Pseudocode:

1	f_min = min(f_min_Sender, f_min_Empfänger)
	f_max = min(f_max_Sender, f_max_Empfänger)
	B_basis = f_max - f_min
	if B_basis <= 0:
5	C_max = 0
	else:
	FSPL = 4 * Pi * d * f_max / c
	B = Eb/N0 * Margin * G/T * EIRP / k_B / FSPL
	B = min(B_basis, B)
10	C_max = sqrt(B)

Abbildung 1 - Algorithmus zur Berechnung der maximalen Datenübertragungsrate in Pseudocode

2.2 Benötigte Daten

Die im Algorithmus verwendeten Größen lassen sich in die Kategorien benötigte Größen, berechnete Größen und Konstanten unterteilen. Aus den berechneten Größen ergibt sich, welche Daten für die Anwendung benötigt werden.

2.2.1 Antennenparameter des Senders

Die Leistungsparameter des Senders sind sein Frequenzbereich und seine äquivalente isotrope Strahlungsleistung. Es wird angenommen, dass der Frequenzbereich des Satelliten nur aus einem einzelnen Downlink besteht, der durch minimale und maximale Frequenz begrenzt ist. Die Grenzfrequenzen und Strahlungsleistung werden vom Anwender vorgegeben.

2.2.2 Antennenparameter des Empfängers

Bei der Antenne der Bodenstation werden ihr Frequenzbereich und ihre Empfangsleistung in Form von G/T für die Berechnung der Datenübertragungsrate bereitgestellt. Dabei kann eine Antenne über mehrere Downlinks in verschiedenen Frequenzbändern verfügen, die jeweils eine minimale und maximale Frequenz besitzen. Die Daten der Bodenstationsantenne werden aus der Datenbank entnommen.

2.2.3 Abstand von Bodenstation und Satellit

Der Abstand zwischen der Antenne einer Bodenstation und dem Satelliten zu einem bestimmten Zeitpunkt wird aus den Positionen beider ermittelt. Die feste Position der Antenne auf der Erdoberfläche kann in der Datenbank gespeichert werden. Da sich der Satellit auf seinem Orbit um die Erde bewegt, hängt die Position des Satelliten in seinem Orbit von der Zeit ab und kann aus den Bahndaten des Satelliten berechnet werden. Die Bahn des Satelliten kann vom Nutzer im NASA/NORAD *Two Line Elements Format* (TLE) eingegeben werden. Mithilfe eines Propagationsmodelles

kann aus diesem die Position des Satelliten zu einem beliebigen Zeitpunkt ermittelt werden (Hoots & Roehrich, 1988).

2.2.4 Signal-Rausch-Verhältnis

Die Angaben bezüglich des für die Datenübertragung mindestens benötigten Signal-Rausch-Verhältnisses werden in Form der Größen E_b/N_0 und Margin getroffen. Die standardmäßigen Annahmen hierfür wurden aus der Link-Budget-Analyse des GSOC übernommen.

Abbildung 2- Tabelle mit übernommenen Werten zur Berechnung der Datenübertragungsrate

Größe	Wert
benötigte Margin	3 dB
benötigte E_b/N_0	2.5 dB

2.3 Verwendete Technologien

2.3.1 Django

Grundlage für diese Besondere Lernleistung bildet das in der populären funktionalen und objektorientierten Programmiersprache Python geschriebene quelloffene Webframework Django (Django Software Foundation, 2020), was 2005 unter einer Open-Source-Lizenz veröffentlicht wurde. Zur Entwicklung der Anwendung wird Djangos integrierter lokaler Entwicklungsserver verwendet und ein neues Django-Projekt angelegt. Dieses enthält die eigentliche Anwendung in Form einer Django-App. Mit Django wurden außerdem die Integration des Datenbanksystems und die Abwicklung aller Datenbankabfragen realisiert. Die Datenbank selbst ist in diesem Fall mit SQLite3 umgesetzt, welches standardmäßig in Django integriert ist (Django Software Foundation, 2020). Django funktioniert grundsätzlich nach dem sogenannten Model-View-Template Muster: Die Anfrage des Nutzers wird von Django entsprechend der angefragten URL an das entsprechende View weitergegeben, welches die Anfrage bearbeitet. Das View kann über das Datenmodell mit der Datenbank interagieren oder mittels des Django Template-Systems ähnlich zu PHP

serverseitig dynamisch HTML- und JavaScript-Code erstellen. Die in dieser Arbeit entwickelte Django-Applikation heißt `orbitalcalc`.

2.3.2 Skyfield

Skyfield ist eine in Python geschriebene und mit Open-Source-Lizenz veröffentlichte Softwarebibliothek zur Berechnung der Positionen von Himmelskörpern und Satelliten (Rhodes, 2020). Es wird zur Berechnung der Position des Satelliten mithilfe des TLEs genutzt.

2.3.3 HTML

Zur Entwicklung der Webanwendung wird die Auszeichnungssprache Hypertext Markup Language (HTML) verwendet. HTML wird als Kernsprache des World Wide Webs zur Strukturierung der grafischen Benutzeroberfläche im Browser genutzt.

2.3.4 CSS

Im World Wide Web ebenfalls essentiell ist die Stylesheet-Sprache Cascading Style Sheets (CSS), welche die visuelle Gestaltung der in der Webanwendung verwendeten HTML-Inhalte ermöglicht.

2.3.5 Java Script

Für Benutzerinteraktionen in der Webanwendung wird die Skriptsprache JavaScript verwendet. Sie ermöglicht das Auswerten von Eingaben und das Verändern und Generieren von Inhalten und erweitert die Möglichkeiten von HTML und CSS.

2.3.6 Cesium

Das in JavaScript geschriebene Open-Source Framework Cesium wird zur dreidimensionalen Darstellung der Antennen, Bodenstationen und der Satellitenlaufbahn verwendet. Zusätzlich werden die Kontakte zwischen Bodenstationen und Satellit visualisiert.

3 Die Datenbank

3.1 Entitäten und Beziehungen

In der Datenbank sollen die Entitäten Bodenstation, Antenne, Link und Betreiber modelliert werden. Ein Link gehört zu genau einer Antenne und eine Antenne kann genau einer Bodenstation zugeordnet werden. Eine Antenne kann mehrere Links besitzen und eine Bodenstation mehrere Antennen. Daraus ergibt sich eine 1:n Beziehung zwischen Bodenstation und Antenne und zwischen Antenne und Link. Ein Betreiber kann außerdem mehrere Bodenstationen betreiben und eine Bodenstation kann zu mehreren Betreibern gehören, woraus sich eine m:n Beziehung zwischen Bodenstation und Betreiber ergibt.

3.2 Attribute

Die Attribute der Relationen ergeben sich aus den für die Berechnung der Datenübertragungsmengen benötigten Daten. Zudem wurden zusätzliche Attribute aufgenommen, die eine Erweiterung der Anwendung in der Zukunft ermöglichen.

3.2.1 Link

Ein Link wird durch die Attribute minimale und maximale Frequenz des Bereiches, welchen er abdeckt, und die Information, ob es sich um einen Downlink oder Uplink handelt, beschrieben. Seine Antenne wird ebenfalls in einem Attribut gespeichert.

3.2.2 Antenne

Eine Antenne besitzt einen Namen und die Positionsdaten sowie Antennenparameter. Die Position ist mit den Attributen Länge, Breite und Höhe über dem Meeresspiegel gegeben. Die Antennenparameter sind in Form von Sendeleistung EIRP, Empfangsleistung G/T, Polarisierung und Durchmesser gegeben. Die Empfangsleistung kann eine Betrachtung von Uplinks, und die Polarisierung eine genauere Berechnung von

Datenübertragungsarten ermöglichen. Ein Attribut beschreibt die Einsatzfähigkeit der Antenne und ermöglicht somit das temporäre Entfernen einer Antenne aus der Auswahl in der Webanwendung. Auch die Bodenstation der Antenne wird in Form eines Attributes gespeichert.

3.2.3 Bodenstation und Betreiber

Bodenstation und Betreiber besitzen als einziges Attribut ihren Namen.

3.3 Entity-Relationship-Modell

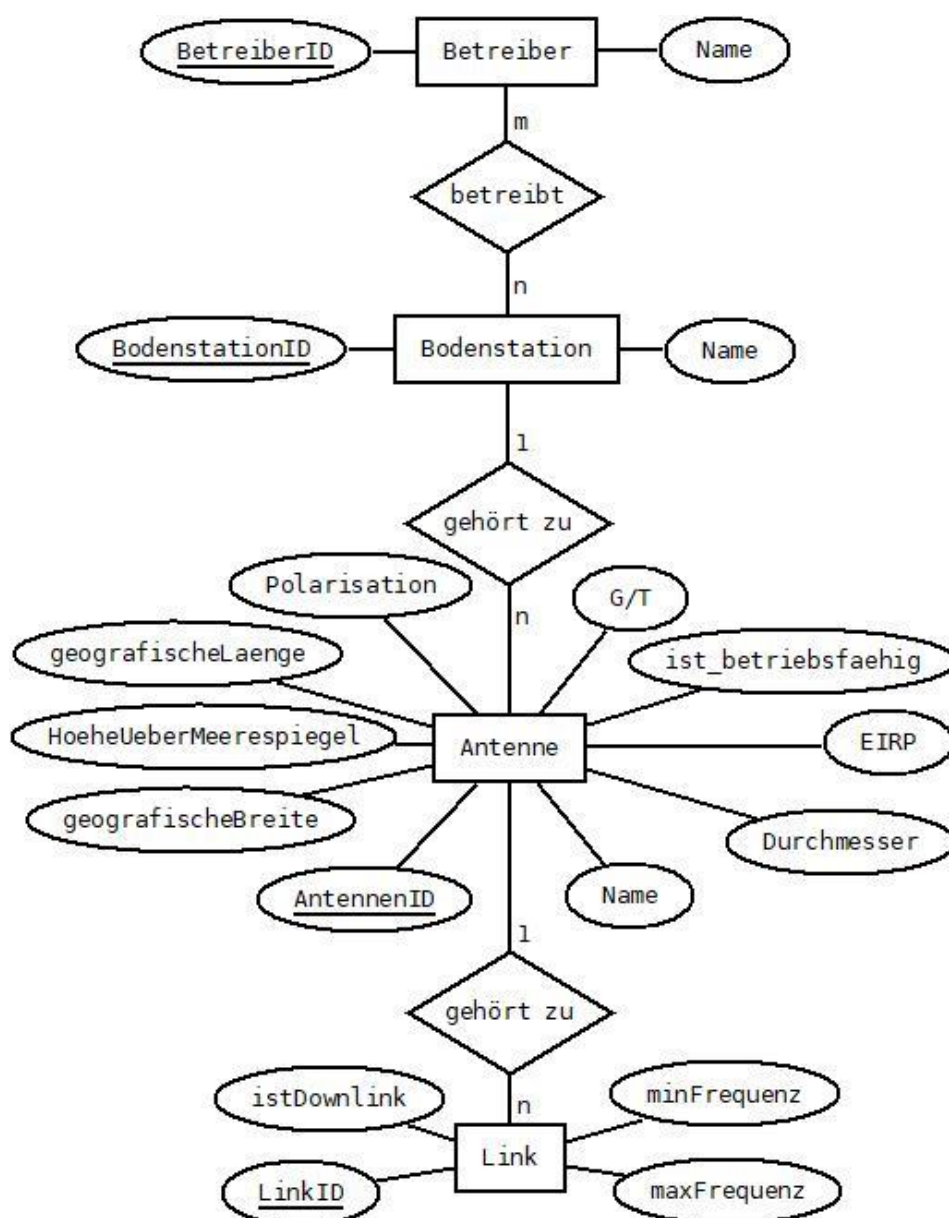


Abbildung 3 - ERD der Datenbank in Chen-Notation

3.4 Implementierung in Django

Eine Relation wird in Django durch eine Python Klasse, genannt Model, implementiert.

```
1 from django.db import models
  # Datenmodell des Links
  class Link(models.Model):
    ...
```

Quelltextbeispiel 1 – Django Model Klasse Link

Die Attribute der Relation werden als Attribute der Model-Klasse definiert und sind dabei selbst Instanzen der sogenannten Model-Feldklassen von Django. Die Art der Feldklasse gibt den Typ eines Attributes an, wobei die bei der Erzeugung der Instanz einer Feldklasse übergebenen Parameter den Wertebereich des Attributs bestimmen. Im folgenden Beispiel wird der Downlinkfrequenz eine Instanz der Feldklasse `DecimalField` zugeordnet. Durch die beim Erstellen der Instanz übergebenen Parameter wird die maximale Anzahl der Stellen auf 10 und die der Nachkommastellen auf 3 festgelegt.

```
1 class Link(models.Model):
    frequency_min_MHz = models.DecimalField(
        max_digits=10,
        decimal_places=3
5    )
```

Quelltextbeispiel 2 – Django Model mit Attribut vom Typ `DecimalField`

Django erlaubt die Definition von abgeleiteten Attributen durch, mit `@property` gekennzeichnete Funktionen der Model-Klasse. Im folgenden Beispiel kann die Funktion `frequency_min` wie ein Attribut ausgegeben werden und gibt die in MHz gespeicherte Frequenz in Hz zurück.

1	MEGA_PREFIX_SIZE = 1e6
	class Link(models.Model):
	frequency_min_MHz = models.DecimalField(
5	max_digits=10,
	decimal_places=3
)
	@property
	def frequency_min(self):
10	return float(self.frequency_min_MHz) \
	* MEGA_PREFIX_SIZE

Quelltextbeispiel 3 – abgeleitetes Attribut in Django Model

Eine Besonderheit stellen Fremd- und Primärschlüssel dar. Solange durch die Feldklasse kein Primärschlüssel definiert ist, wird von Django automatisch eine fortlaufende Ganzzahl als Primärschlüssel mit dem Bezeichner `id` verwendet. Für Fremdschlüssel muss das entsprechende Attribut eine Instanz der Klasse `django.models.ForeignKey` sein.

1	class Link(models.Model):
	...
	aperture = models.ForeignKey(
	# Fremdschlüssel verweist auf die Relation der Antenne
5	Aperture,
	# Link wird gelöscht, wenn seine Antenne gelöscht wird
	on_delete=models.CASCADE,
)

Quelltextbeispiel 4 – Fremdschlüssel im Django Model

Beziehungen vom Typ `m:n` werden durch die Klasse `models.ManyToManyField` in einer der beiden involvierten Relationen definiert. Dadurch müssen keine Verknüpfungsrelationen erstellt werden.

1	# Model der Bodenstation
	class GroundStation(models.Model):
	# Name als String
	name = models.CharField(max_length=200)
5	# Many-to-Many Beziehung zu Betreibern
	operator = models.ManyToManyField(Operator)
	...

Quelltextbeispiel 5 - m:n Beziehung in Django

Zur Anwendung von Änderungen am Datenmodell auf die Datenbank, können sogenannte Migrationen erstellt und ausgeführt werden. Die Migrationen bauen chronologisch aufeinander auf. In ihnen sind die Änderungen am Datenmodell gegenüber der vorhergehenden Migration aufgeführt. Nach der Bearbeitung eines Datenmodells können Migrationen mit Django automatisch, aber auch manuell, erstellt und auf die Datenbank angewandt werden.

3.5 Datenbankabfragen

Datenbankabfragen werden mit der Django Datenbank API möglich. Beim Zugriff auf alle Einträge einer Relation über das Attribut `objects` erhält man eine Instanz der Klasse `QuerySet`. Durch das Anwenden von Filtern wird auf spezifische Elemente zugegriffen.

```
1 links = Link.objects
  # Alle Links
  all_links = links.all()
  # Links nach spezifischem Attribut filtern
5 links_of_specific_antenna = \
    links.filter(aperture_id=some_antenna_id)
```

Quelltextbeispiel 6 - Datenbankabfragen

3.6 Population mit Daten

Die Datenbank wurde mit vom DLR zur Verfügung gestellten Daten in Form einer Excel Tabelle des GSOC angereichert. Die Excel Datei mit dem Namen `121115_ground_stations_list.xlsx` befindet sich auf der beigelegten CD. Sie wurde mit Hilfe der Python Softwarebibliothek eingelesen. Die in verschiedenen Formaten gegebenen Daten wurden mit dafür geschriebenen Scripts umgewandelt. Datensätze mit einzelnen besonders formatierten Einträgen wurden manuell abgeändert. Unvollständige Daten oder solche, die mit Fragezeichen als ungewiss markiert waren, wurden nicht übernommen. Dadurch ergaben sich unvollständige Datensätze, welche zwar für die Anwendung nicht nutzbar sind, aber in Zukunft ergänzt werden können. Dementsprechend wurden die Model-Klassen um eine Funktion ergänzt, welche im Falle der

Bodenstation nur vollständige Antennendatensätze beziehungsweise für eine Antenne nur vollständige Linkdatensätze zurückgibt, die für die Berechnungen verwendet werden.

```
1 class Link(models.Model):
    ...
    # prüft, ob alle Attribute Werte besitzen
    @property
5     def is_usable(self):
        return self.frequency_max_MHz \
               and self.frequency_min_MHz \
               and self.is_downlink is not None
```

Quelltextbeispiel 7 – Funktion zur Prüfung, ob alle Attribute eines Eintrags gültig sind

```
1 class Aperture(models.Model):
    ...
    @property
    # gibt liste von nutzbaren Links zurück
5     def usable_links(self):
        links = Link.objects.filter(aperture_id=self.id)
        usable = list()
        for link in links:
            if link.is_usable:
10             usable.append(link)
        return usable
```

Quelltextbeispiel 8 - Rückgabe der vollständigen Datensätze der zu einer Antenne gehörigen Links

3.7 Zugriff auf die Datenbank durch den Nutzer

Das manuelle Anzeigen, Pflegen und Hinzufügen von Datensätzen ist in Form der Adminseite in Django integriert und ermöglicht den Nutzern die Datenbank auf dem neusten Stand zu halten. Dafür können standardmäßig Nutzern unterschiedliche Rechte vergeben werden.

4 Die Anwendung

4.1 Grundlegende Funktionalität

Die Grundfunktionalität der Webapplikation ist die Ermittlung der optimalen Abfolge von Kontakten zwischen Satellit und Bodenstationen zum

Erreichen der maximal übertragbaren Datenmenge Der Zeitraum der Betrachtung der Datenübertragung der Satellitenmission wird dabei als Analysezeitraum bezeichnet. Vom Anwender werden die Daten des Satelliten eingegeben und es wird eine Auswahl von Bodenstationsantennen getroffen. Des Weiteren werden die Ergebnisse mit einer für den Missionszeitraum angestrebten Datenübertragungsmenge verglichen. Es wird nur die über den gesamten Analysezeitraum übertragbare Datenmenge in dieser Anwendung in Betracht gezogen. Diese Anwendung ist also nicht für Missionen geeignet, bei denen Daten innerhalb eines strikten Zeitraumes nach Ihrer Aufnahme zurück zur Erde gesendet werden müssen. Das ist besonders bei Kommunikationssatelliten der Fall. Bei wissenschaftlichen Forschungsmissionen, auf welche die Anwendung ausgerichtet sein soll, ist das allerdings seltener der Fall.

4.1.1 Analysemodi

Es wurde sich für vier Kriterien entschieden, nach denen die Antennen aufgeteilt werden. Diese Kriterien werden im Folgenden Analysemodi genannt. Der erste Analysemodus heißt „Antennen“ und betrachtet jede ausgewählte Antenne und die über sie übertragbare Datenmenge einzeln. Der Analysemodus „Bodenstationen“ betrachtet für jede Bodenstation die mit allen Antennen maximal übertragbare Datenmenge. Analog wird im Analysemodus „Betreiber“ jeweils für jeden ausgewählten Betreiber die maximal über alle ausgewählten Antennen aller ausgewählten Bodenstationen übertragbare Datenmenge ermittelt. Der Analysemodus „alle“ betrachtet die über alle ausgewählten Antennen übertragbare Datenmenge.

4.2 Django Template-System

Ein View in Django ist eine Python-Funktion, welche eine HTTP Anfrage als Argument nimmt und eine Antwort zurückgibt. Diese Antwort kann beispielsweise ein HTML-Dokument, eine Weiterleitung, eine Datei oder ein Fehler sein (Django Software Foundation, 2020). Django ermöglicht es damit HTML und Python Code zu separieren. Der Python Code steht im

View und der HTML Code im Template. Die Funktion `render()` kombiniert die im Dictionary context übergebenen Werte mit dem Template. Das Dictionary mit den übergebenen Werten wird im Folgenden als Kontext bezeichnet.

```
1  from django.shortcuts import render
   # View-Funktion
   def hello_view(request):
5      template_name = "app_name/hallo.html"
      # zu übergebende Parameter
      context = {
          "greeting": "Hallo Welt!",
          "animals": [
10         "Wurm",
          "Otter",
          "Giraffe"
        ]
      }
15  return render (request, template_name, context)
```

Quelltextbeispiel 9 - Django View

Im Template wird die sogenannte Django Template Language verwendet. Mit `{{ variable_name }}` markierte Variablen aus dem Kontext und mit `{% tag_name % }` markierte Logikbausteine werden von Django erkannt und interpretiert.

```
1  <!DOCTYPE html>
   <html>
      <head>
          <title>Begrüßung</title>
5      </head>
      <body>
          Begrüßung: {{ greeting }}<br>
          Tiere:
          # Zählschleife
10     {% for animal in animals %}
          <br>- {{ animal }}
          {% endfor %}
      </body>
   </html>
```

Quelltextbeispiel 10 - Django Template

Das Ergebnis des Templates mit den übergebenen Daten aus den oberen beiden Beispielen ist eine HTML-Datei.

Begrüßung: Hallo Welt!

Tiere:

- Wurm
- Otter
- Giraffe

Abbildung 4 - Ergebnis des Django Template und View

Anfragen werden dem entsprechenden View mit sogenannten urlpatterns in der Datei `urls.py` zugeordnet.

```
1 from django.conf.urls import patterns, url  
  
   view = app_name.views.hello  
  
5 urlpatterns = [  
    url("greeting/", view, name = "greeting")  
]
```

Quelltextbeispiel 11 – Django URL Mapping

4.3 Eingabe

4.3.1 Eingabewerte

Die Eingabe für eine Analyse in der Webapplikation kann in Satellitendaten, Analyseparameter und Auswahl der Antennen, Bodenstationen und Betreiber unterteilt werden.

4.3.1.1 Satellitendaten

Satellitendaten

Two Line Element Set

```
ISS (ZARYA)
1 25544U 98067A   20196.81549769 -.00000199  00000-0  44991-5 0  9999
2 25544   51.6443 211.7288 0001419 115.0512 224.2366 15.49514614236291
```

EIRP dBW

Downlinkfrequenzen

Von GHz

bis GHz

Abbildung 5 - Eingabebereich der Satellitendaten

Die Satellitendaten bestehen aus dem TLE sowie den Antennenparametern mit EIRP und minimaler und maximaler Downlinkfrequenz. Dabei kann bei der EIRP zwischen den Einheiten Watt (W) und den im Kontext von Link Budget Analysen verwendeten Dezibel Watt (dBW) gewählt werden. Die Frequenzen werden in Herz angegeben, wobei aufgrund der großen Zahlen in unterschiedlichen Größenordnungen zwischen Dezimalpräfixen des Internationalen Einheitensystems gewählt werden kann.

4.3.1.2 Analysedaten

Datenübertragung

angestrebte Datenübertragungsmenge

MB

pro benutzerdefiniert

Tage

Stunden

Minuten

Betrachtung von

Bodenstationen einzeln

Zeitraum

gesamter Analysezeitraum (UTC)

Startzeit

Endzeit

Abbildung 6 - Eingabebereich der Analysedaten

Die Analysedaten umfassen Analysezeitraum, Analysemodus und angestrebte Datenmenge. Der Analysezeitraum wird mit Datum und Uhrzeit von Start- und Endzeitpunkt eingegeben.

Aus einer Liste kann der Nutzer den Analysemodus wählen.

Die zu übertragende Datenmenge wird zusammen mit ihrer Einheit eingegeben.

Ausgehend davon, dass Nutzlastdaten eines Satelliten zeitlich periodisch anfallen können, wird die Möglichkeit mehrerer Eingabeoptionen gegeben. Es kann ausgewählt werden, ob die eingegebene Datenmenge über den ganzen Analysezeitraum anfällt, pro Orbit des Satelliten, oder in einem benutzerdefinierten Zeitraum. Dieser benutzerdefinierte Zeitraum kann in Form von Tagen, Stunden und Minuten bei Auswahl des benutzerdefinierten Zeitraumes eingegeben werden.

4.3.1.3 Auswahl der Antennen, Bodenstationen und Betreiber

Bodenstationen

- Allan Park
- Aussaguel
- Bangalore
- Belrose
- Biak
- Carpentersville
- Castle Rock
- Clarksburg
- Dongara
- Fillmore
- Fucino
- Gnangara
- Hartebeesthoek
- Hassan
- Kerguelen
- Kiruna
- Kourou
- Kumsan
- Perth
- Uralla
- FMA-1
- FMA-2
- Weilheim

Betreiber

- CNES
- DLR
- ESA
- ISRO
- JAXA
- NASA
 - Clarksburg
 - Carpentersville
 - Allan Park
 - Castle Rock
- Prioranet
- Telespazio
- USN

Antennen der Bodenstation anzeigen/ausblenden

Abbildung 7 - Eingabebereich Antennen, Bodenstationen und Betreiber

Die Auswahl der Betreiber, Bodenstationen und Antennen erfolgt jeweils nach Namen aus einer Liste. Die Beziehungen zwischen ihnen sind in 3.1 Entitäten und Beziehungen beschrieben. Daraus lässt sich ableiten, dass es Kombinationen von Betreibern, Bodenstationen und Antennen gibt, die sich logisch ausschließen. Beispielsweise kann eine Bodenstation keine Daten übertragen, wenn keine ihrer Antennen ausgewählt wurde. Im Umkehrschluss muss eine Bodenstation ausgewählt sein, wenn eine ihrer Antennen ausgewählt wurde. Bereits bei der Eingabe werden derartige unmögliche Kombinationen nicht zugelassen, indem bei der Auswahl der ersten Antenne einer Bodenstation die Antenne entsprechend automatisch ausgewählt wird. Dementsprechend werden alle ausgewählten Antennen bei der Abwahl einer Bodenstation ebenfalls abgewählt. Die m:n Beziehung zwischen Bodenstationen und Betreibern wurde umgesetzt, indem beim

Abwählen eines Betreibers alle Bodenstationen, von denen nur dieser Betreiber ausgewählt war, ausgewählt werden. Wird ein Betreiber gewählt, werden alle seine Bodenstationen ausgewählt. Analog werden die Betreiber einer Bodenstation ausgewählt, wenn ihre einzige ausgewählte Bodenstation ausgewählt wird und alle Betreiber einer Bodenstation werden beim Anwählen einer Bodenstation ausgewählt.

4.3.2 Umsetzung

4.3.2.1 Eingabe mit Django Forms

Ähnlich zu den Models ermöglicht Django das Erstellen von Forms als Python-Klassen. Die Attribute der Klasse repräsentieren ein Eingabefeld, wobei der zugewiesene Wert den Typ des Eingabefeldes festlegt. Die den Attributen zugewiesenen Werte sind Instanzen der Form Feldklassen von Django. Die beim Erstellen der Attribute übergebenen Werte legen den Wertebereich fest und geben durch das sogenannte Widget die Umsetzung des Feldes in HTML an.

```
1  from django.forms import Form, DecimalField, NumberInput
   class InputForm(Form):
       # Feldklasse -> Dezimalzahl
5   data = DecimalField(
       # Widget -> HTML input type number
       widget=NumberInput(
           # Ausschluss der Eingabe negativer Zahlen
           attrs={'min': 0}
10      )
   )
```

Quelltextbeispiel 12 - Django Form Grundlagen

Bei einem Auswahlfeld können die Optionen in Form des Parameters `choices` übergeben werden. Die Optionen sind in diesem Fall eine Liste bestehend aus 2-Tupeln.

1	<code>from django.forms import Form, DecimalField, ChoiceField, NumberInput, Select</code>
5	<code>DATA_UNITS = [</code> <code> (1, "bit"),</code> <code> (8, "Byte"),</code> <code> (8e3, "kB"),</code> <code> (8e6, "MB"),</code> <code> (8e9, "GB"),</code> <code> (2 ** 10, "KiB"),</code> <code> (2 ** 20, "MiB"),</code> <code> (2 ** 30, "GiB")</code> <code>]</code>
15	<code>class InputForm(Form):</code> <code> data = DecimalField(...)</code> <code> data_unit = ChoiceField(</code> <code> # Auswahlmöglichkeiten</code> <code> choices=DATA_UNITS,</code> <code> widget=Select()</code> <code>)</code>
20	

Quelltextbeispiel 13 - Django Form Auswahlfeld

Eine Instanz der Formklasse wird beim Rendern des Templates im Kontext übergeben.

1	<code>from django.shortcuts import render</code> <code>from app_name.forms import InputForm</code>
5	<code>def input_view(request):</code> <code> template_name = "app_name/input.html"</code> <code> # Erstellen der Instanz der Formklasse</code> <code> input_form = InputForm()</code> <code> context = {</code> <code> "form": input_form</code> <code> }</code>
10	<code> return render (request, template_name, context)</code>

Quelltextbeispiel 14 - Erstellen von Django Form im View

1	<!DOCTYPE html>
	<html>
	<head>
	<title>Eingabe</title>
5	</head>
	<body>
	<form>
	<!-- Datenmenge -->
	{{ form.data }}
10	<!-- Einheit -->
	{{ form.data_unit }}
	<!-- Fehler -->
	{{ form.data.errors }}
	{{ form.data_unit.errors }}
15	</form>
	</body>
	</html>

Quelltextbeispiel 15 - Django Form im Template

Wie im letzten Beispiel kann Django Forms auch durch das Überprüfen der Gültigkeit eines Eingabewertes übernommen werden. Dafür kann die Funktion `clean()` der Parent-Klasse überschrieben werden. Im folgenden Beispiel wird überprüft, ob die Startzeit des Analysezeitraumes vor der Endzeit liegt.

1	<code>class InputForm(Form):</code>
	<code>...</code>
	<code>def clean(self):</code>
	<code>cleaned_data = super().clean()</code>
5	<code># Startzeit < Endzeit pruefen</code>
	<code>start_time = cleaned_data.get("start_time")</code>
	<code>end_time = cleaned_data.get("end_time")</code>
	<code>if not start_time < end_time:</code>
	<code>error_message = "Startzeit nach Endzeit"</code>
10	<code>self.add_error('end_time', error_message)</code>

Quelltextbeispiel 16 - Überprüfung der Gültigkeit eines Django Forms

4.3.2.2 Eingabe des TLEs mit Regular Expression

Besonders wichtig ist das Überprüfen der Eingabe auf Korrektheit beim TLE. Die Satellitenbahnelemente werden als Zeichenblöcke in zwei Zeilen

dargestellt, wobei die Position und Bedeutung der Zeichenblöcke festgeschrieben sind.

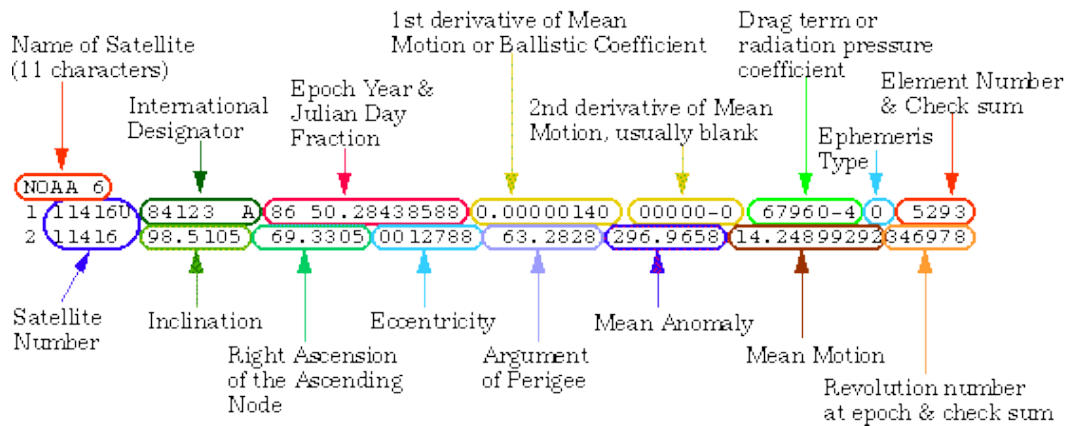


Abbildung 8 - Aufbau eines TLE

https://spaceflight.nasa.gov/realdata/sightings/SSApplications/Post/JavaSSOP/SSOP_Help/tle_def.html

Zugriff: 07.07.2020

Entsprechend der obigen Abbildung wurde ein regulärer Ausdruck erstellt, mit welchem eine Zeichenfolge darauf geprüft werden kann, ob sie ein gültiges TLE darstellt.

```

1  # Regex Ausdruck laut NASA
   REGEX_TLE = str()
   # Satellitenname
   REGEX_TLE += ".{0,24} {0,45}\r{0,1}\n"
5  # erste Zeile
   REGEX_TLE += "1 " # Zeilennummer
   REGEX_TLE += "[0-9]{5}" # Katalognummer
   ...
   """ Rest siehe regex_tle.py """
15
   NUMBER_OF_ROWS_TLE = 2
   NUMBER_OF_COLUMNS_TLE = 69
   MAX_LENGTH_TLE = 24 + NUMBER_OF_COLUMNS_TLE * NUMBER_OF_ROWS_TLE + 4

```

Quelltextbeispiel 17 - Regulärer Ausdruck TLE

Der reguläre Ausdruck wurde mit TLEs von CelesTrak (celestrak.com) auf Korrektheit überprüft. Außerdem wurden in TLEs testweise einzelne Zeichen falsch abgeändert und geprüft, ob der eingebaute Fehler erkannt wird. Das Überprüfen der Eingabe erfolgt automatisch über die Feldklasse

RegexField, welcher der reguläre Ausdruck in Form des Strings REGEX_TLE übergeben wird.

```
1 from django.forms import Form, RegexField
  from orbitalscalc.regex_tle import REGEX_TLE,
    MAX_LENGTH_TLE, NUMBER_OF_COLUMNS_TLE,
    NUMBER_OF_ROWS_TLE
5
  class InputForm(Form):
    tle = RegexField(regex=REGEX_TLE, widget=Textarea(attrs={
      # Maximalanzahl Zeichen
      # + 2 Zeichen für Zeilenumbruch
10      # und 2 Zeichen für eventuellen Wagenrücklauf
      "maxlength": MAX_LENGTH_TLE,
      "rows": NUMBER_OF_ROWS_TLE,
      "cols": NUMBER_OF_COLUMNS_TLE,
      # manuelles Ändern von Größe verhindern
15      "style": "resize:none"
    )))
```

Quelltextbeispiel 18 - RegexField TLE

4.3.2.3 Anzeigen der Bodenstationen in Cesium



Abbildung 9 - Ausschnitt aus der Darstellung in der Webanwendung mit Cesium. Die Bodenstation Weilheim ist ausgewählt.

Die Darstellung auf dem dreidimensionalen Globus ist für die Anwendung nicht essentiell, bietet zugleich aber die Möglichkeit die Ergebnisse der Webanwendung einfacher zu interpretieren und mögliche Unplausibilitäten

oder Fehler besser zu erkennen. Da Cesium für die Anwendung nicht von essentieller Wichtigkeit ist, werden CSS und JavaScript Bibliotheken auf dem Server der Anwendung nicht lokal gespeichert, sondern von der Cesium-Website importiert. Das `<div>` Element mit der ID `cesiumContainer` enthält den Viewer, in welchem die dreidimensionale Darstellung stattfindet. Um auf detailliertes Kartenmaterial zuzugreifen, kann ein Nutzer sich auf der Cesium-Website für nicht kommerzielle Zwecke kostenlos registrieren und einen Zugangsschlüssel für das Kartenmaterial von Bing erhalten. Ein solcher Zugangsschlüssel wurde für diese Arbeit erworben.

1	<code><!-- Cesium --></code>
	<code><!-- Reihenfolge von Elementen wichtig --></code>
	<code><script src="..."></script></code>
	<code><link href="..." rel="stylesheet"/></code>
5	<code><!-- enthält Visualisierung --></code>
	<code><div id="cesiumContainer"></div></code>
	<code><script></code>
	<code> // Zugangsschlüssel für Datenmaterial</code>
	<code> Cesium.Ion.defaultAccessToken = 'your_access_token';</code>
10	<code> var viewer = new Cesium.Viewer('cesiumContainer');</code>
	<code></script></code>

Quelltextbeispiel 19 - Cesium Grundlagen

Zu dem Viewer können nun Objekte der Klasse `Cesium.Entity` hinzugefügt werden. Im folgenden Quelltextbeispiel 20 wird für jede Bodenstation ein Entity mit Icon, Beschriftung und Position auf der Karte erstellt.


1	<code><script></code>
	<code>const PREFIX_GROUND_STATION = "ground_station"</code>
	<code>// Darstellung der Bodenstationen</code>
	<code>{% for ground_station in</code>
5	<code>display_information.ground_stations.values %}</code>
	<code>// für Bodenstation Cesium Objekt erstellen</code>
	<code>var ground_station = new Cesium.Entity({</code>
	<code>id: PREFIX_GROUND_STATION</code>
	<code>+ {{ ground_station.id }},</code>
10	<code>name: "{{ ground_station.name }}",</code>
	<code>// Icon an Position der Bodenstation</code>
	<code>billboard: {...},</code>
	<code>// Beschriftung der Bodenstation mit Name</code>
	<code>label:{...},</code>
15	<code>// Position der Bodenstation</code>
	<code>position: Cesium.Cartesian3.fromDegrees(</code>
	<code>{{ ground_station.lon }},</code>
	<code>{{ ground_station.lat }}</code>
	<code>)</code>
20	<code>});</code>
	<code>// Cesium.Entity Objekt zu Viewer hinzufügen</code>
	<code>viewer.entities.add(ground_station)</code>
	<code>{% endfor %}</code>
	<code></script></code>

Quelltextbeispiel 20 - Erstellen des Cesium Entity für Bodenstation

Eine in der Eingabeliste ausgewählte Bodenstation besitzt ein oranges Icon und eine türkis hervorgehobene Beschriftung, wohingegen eine nicht gewählte Bodenstation ausgegraut ist. Aus Gründen der Übersichtlichkeit und um Überlappungen zu verhindern, werden die Beschriftungen ab einem Abstand von der Kamera von 8.000 km ausgeblendet.

Analog zu den Bodenstationen werden die Antennen zum Viewer hinzugefügt. Beim Hineinzoomen werden die Bodenstationen unter einer Entfernung von 4 km von der Kamera nicht mehr angezeigt. Anstelle der Bodenstation werden ihre einzelnen Antennen eingeblendet.

Der Nutzer kann durch Gedrückthalten und Ziehen der Maus die Kamera um den Globus bewegen. Zoomen erfolgt durch Drehen des Mausekkrades. Das Neigen der Kamera wurde zur Erleichterung der Navigation deaktiviert.

Mit einem Klick auf den Positionspinn  neben dem Namen einer Antenne oder Bodenstation in der Auswahlliste fliegt die Kamera auf der Karte zu dieser hin.

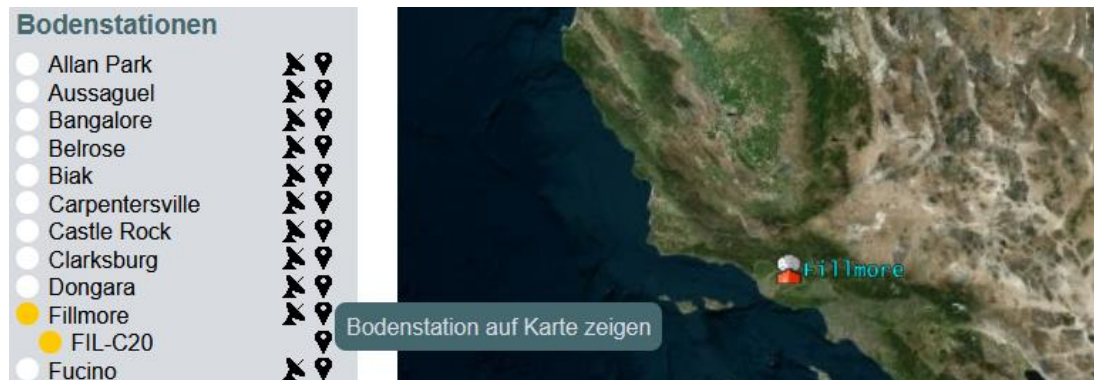


Abbildung 10 - Anzeigen einer Bodenstation auf der Karte

Alle Icons und Piktogramme, welche nicht standardmäßig Teil des Cesium-Viewers sind, wurden selbst erstellt. Zur eindeutigen Verständlichkeit wurden zu den Schaltflächen kleine Pop-up-Fenster mit einem erklärenden Text hinzugefügt.

4.4 Analyse Datenübertragung

4.4.1 Ermittlung der Kontakte

Jeder der Analysemodi läuft auf das gleiche mathematische Problem hinaus, und zwar die Ermittlung der maximal möglichen Datenübertragungsmenge mit einer gegebenen Auswahl an Antennen. Dazu werden für jede Antenne die Zeitspannen ermittelt, in welchen sich der Satellit aus Sicht der Antenne über dem Horizont befindet. Eine solche Zeitspanne wird als Kontakt bezeichnet. Nur wenn der Satellit aus der Perspektive der Bodenstation nicht von der Erde verdeckt ist, kann eine Funkverbindung aufgebaut werden. Die Datenübertragung ist also nur während eines Kontaktes möglich. Es wird davon ausgegangen, dass ein Funkkontakt zwischen Bodenstation und Satellit ab einer Elevation über dem Horizont der Bodenstation von 5° möglich ist. Diese Annahme zieht die negativen Einflüsse des langen Weges durch die Atmosphäre auf die Funkwellen in Betracht. Die Ermittlung der Kontakte geschieht mit Skyfield, indem aus dem TLE ein Objekt der Klasse EarthSatellite erstellt wird.

Für jede Antenne wird ein Objekt der Klasse Topos mit den Koordinaten der Antenne erstellt.

```
1  from skyfield.api import EarthSatellite

    satellite_skyfield = EarthSatellite(
        tle[LINE_ONE_IN_TLE],
5    tle[LINE_TWO_IN_TLE],
        tle[LINE_OF_NAME_IN_TLE]
    )
    antenna_skyfield = Topos(
        latitude_degrees=float(antenna.latitude),
10    longitude_degrees=float(antenna.longitude),
        elevation_m=float(antenna.altitude)
    )
```

Quelltextbeispiel 21 - Erstellen der Objekte für Satellit und Antenne in Skyfield

```
1  times, events = satellite_skyfield.find_events(
    antenna_skyfield,
    # Start und Ende des Analysezeitraumes
    start_time_skyfield,
5    end_time_skyfield,
    # Mindesthöhe über Horizont bei Kontakt
    altitude_degrees =
        MINIMUM_HEIGHT_OVER_HORIZON_FOR_CONTACT_DEGREES
    )
```

Quelltextbeispiel 22 - Ermitteln der Kontakte mit Skyfield

Als Liste events werden die Ereignisse aus Sicht der Bodenstation in chronologischer Reihenfolge zurückgegeben. Das sind Aufgang, Höchststand und Untergang des Satelliten. Der Höchststand kann auch als Kulmination bezeichnet werden. Ihnen sind von Skyfield in gleicher Reihenfolge die Werte 0, 1 und 2 zugeordnet, welche in einer Enumeration gespeichert werden.

1	<code>from enum import IntEnum</code>
	<code>class SkyfieldEventTypes(IntEnum):</code>
	<code> Rise = 0</code>
5	<code> Culminate = 1</code>
	<code> Set = 2</code>

Quelltextbeispiel 23 - Enumeration für die Skyfield Events

Ein Kontakt besteht in der Regel aus Aufgang, Kulmination und Untergang. Besonders werden die Fälle betrachtet, bei denen der erste Kontakt vor dem Analysezeitraum beginnt oder der letzte Kontakt nach dem Analysezeitraum endet und somit ein unvollständiger Kontakt vorliegt. In diesen Fällen wird als Startzeitpunkt eines unvollständigen Kontaktes am Anfang der Startzeitpunkt des Analysezeitraumes festgelegt. Analog wird mit einem unvollständigen letzten Kontakt verfahren. Zusätzlich muss der Fall betrachtet werden, dass überhaupt kein vollständiger Kontakt zu Stande kommt. Das kann bei geostationären Satelliten der Fall sein kann, welche sich relativ zur Erdoberfläche nicht, beziehungsweise nur wenig, bewegen. Geostationäre Satelliten können auch mehrere nicht durch Auf- oder Untergänge unterbrochene, zusammenhängende Höchststände aufweisen.

Für jedes Ereignis in `events` ist an der gleichen Stelle in der Liste `times` der Zeitpunkt gegeben. Mit den Daten wird ein Objekt der Klasse `RelativePosition` erstellt, welches die Positionsdaten des Satelliten und seine räumliche Beziehung zur Antenne zu einem bestimmten Zeitpunkt repräsentiert. Dazu wird mit Skyfield der Unterschied zwischen den Positionsvektoren von Antenne und Satellit zum entsprechenden Zeitpunkt berechnet.

1	<code># Relative Position von Satellit und Antenne zum Zeitpunkt</code>
	<code>relative_position_skyfield =</code>
	<code> (satellite_skyfield - antenna_skyfield())</code>
	<code> .at(time_skyfield)</code>

Quelltextbeispiel 24 - Berechnung der relativen Position mit Skyfield

Mithilfe von `relative_position_skyfield` lassen sich nun Abstand und Relativgeschwindigkeit von Antenne und Satellit ermitteln, welche zur Berechnung der Datenübertragungsraten benötigt werden. Ein Kontakt zwischen Antenne und Satellit wird durch die Klasse `Contact` repräsentiert, in welchem alle zugehörigen relativen Positionen in chronologischer Reihenfolge gespeichert werden. Bei einem vollständigen Kontakt sind das jeweils eine relative Position für Aufgang, Höchststand und Untergang des Satelliten. Nach der Erstellung aller Kontakte einer Antenne werden diese in chronologischer Reihenfolge zur Kontaktfolge der Antenne hinzugefügt. Die Kontaktfolge ist ein Objekt der Klasse `ContactSequence` und repräsentiert eine chronologische Abfolge von Kontakten.

Bei der Ermittlung der Kontakte muss beachtet werden, dass ein TLE nur einen gewissen Zeitraum von etwa einem Jahr besitzt, in welchem es noch exakt ist. Eine Erweiterung der Anwendung mit der Eingabe mehrerer TLEs, welche zu unterschiedlichen Zeitpunkten gültig sind, ist zukünftig denkbar.

4.4.2 Berechnung der Datenmengen

Nach dem Ermitteln der Kontakte wird für jeden Kontakt die maximal übertragbare Datenmenge berechnet. Dafür wird für jede relative Position mit dem Algorithmus aus 2.1.2 die maximal übertragbare Datenmenge ermittelt und im jeweiligen `RelativePosition` Objekt gespeichert. Zur Berechnung der Datenmengen wird neben den Übertragungsraten der einzelnen relativen Positionen ein Zeitraum benötigt, über welchen die Datenübertragung mit der entsprechenden Datenrate stattfindet. Dafür werden die Mittelpunkte zwischen den Zeiten der relativen Positionen als Grenzen der Zeiträume angenommen. Die erste und letzte relative Position

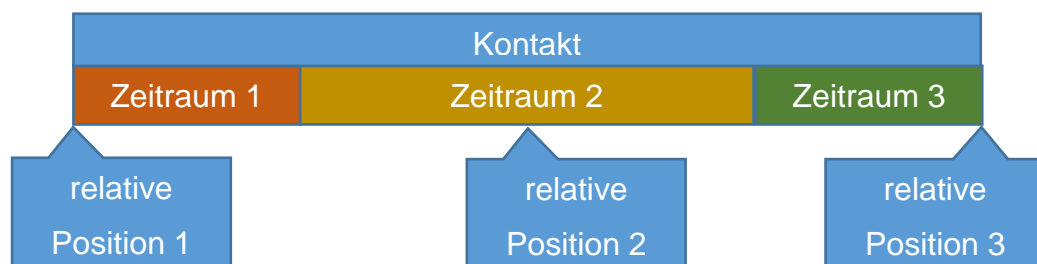


Abbildung 11 - Aufteilung der Zeiträume in einem Kontakt

bilden eine Ausnahme, da sie keinen Vorgänger beziehungsweise Nachfolger besitzen. Der Zeitraum der ersten relativen Position beginnt mit dem Zeitpunkt der ersten relativen Position und der Zeitraum der letzten relativen Position endet mit dem Zeitpunkt der letzten relativen Position.

Die in einem Zeitraum einer relativen Position übertragbare Datenmenge ist das Produkt aus der Länge des Zeitraumes und der Datenrate der relativen Position. Die Datenübertragungsmenge eines Kontaktes ist demnach die Summe der in den einzelnen Zeiträumen übertragbaren Datenmengen. Dabei muss beachtet werden, dass eine Antenne mehrere Links besitzen kann. Es wird also für jeden Link die maximal übertragbare Datenmenge ermittelt und schließlich der Link mit der höchsten Datenübertragungsmenge gewählt.

4.4.3 Ermittlung der optimalen Kontaktfolge

Die Ermittlung der optimalen Kontaktfolgen unterscheidet sich je nach Analysemodus. Im Analysemodus „Antennen“ wird jede Antenne einzeln betrachtet. An einer einzelnen Antenne kann immer nur ein Kontakt gleichzeitig stattfinden, weshalb die Kontakte sich nicht überlappen können. Damit ist die Kontaktfolge einer einzelnen Antenne gleichzeitig die optimale Kontaktfolge, also die Kontaktfolge, mit welcher die größte Datenmenge übertragen werden kann.

In den anderen Analysenmodi wird ebenfalls die Kontaktfolge gesucht, mit welcher die maximale Datenmenge übertragen werden kann. Die Kontakte können dabei von mehreren Antennen stammen. Im Analysemodus „alle“ wird aus allen Kontakten die beste Kontaktfolge gesucht, im Analysemodus „Bodenstationen“ aus allen Kontakten aller Antennen einer Bodenstation und im Analysemodus „Betreiber“ aus allen Kontakten aller Antennen aller Bodenstationen jedes einzelnen ausgewählten Betreibers.

Für Forschungsmissionen werden auf Grund der geringeren Startgebühren für den Satelliten häufig niedrige Umlaufbahnen genutzt. Auf diesen Bahnen sind Kontakte mehrere Sekunden bis wenige Minuten lang. Zur

Kontaktaufnahme mit einer anderen Antenne wird Zeit benötigt, welche einen großen Teil dieser kurzen Zeiträume ausmachen kann. Sobald sich zwei mögliche Kontakte überlappen, muss eine Entscheidung zwischen den beiden Kontakten getroffen werden. Für die Ermittlung der optimalen Kontaktfolgen wird vereinfachend davon ausgegangen, dass ein Kontakt immer vollständig oder gar nicht stattfindet.

Die Ermittlung der besten Kontaktfolge aus einer Menge von Kontakten findet in der Funktion `determine_best_contact_sequence()` in der Datei `orbitalcalc.contact_utility.py` statt. Dazu werden die Kontakte zuerst nach ihrem Startzeitpunkt geordnet. Anhand dessen werden sie in chronologischer Reihenfolge in Gruppen sortiert, zwischen denen keine weiteren Kontakte mehr stattfinden. In diesen Gruppen kommt es zur Überlappung von Kontakten. Die Kontakte einer Kontaktgruppe bleiben dabei chronologisch nach ihrer Startzeit geordnet. Die Ermittlung der sortierten Kontaktgruppen erfolgt mit der Funktion `determine_sorted_contact_groups()`.

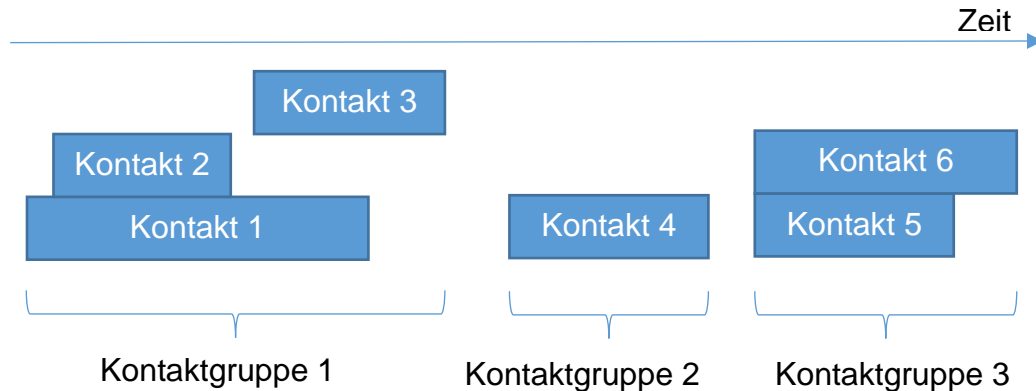


Abbildung 12 - Schematische Darstellung von Kontaktgruppen

Anschließend wird mit der Funktion `best_contact_sequence_from_sorted_group()` für jede Kontaktgruppe die optimale Kontaktfolge ermittelt. Dazu werden mit der Funktion `smallest_overlapping_contact_group()` zuerst alle Kontakte der Gruppe ermittelt, welche mit dem Kontakt überlappen, der zuerst endet. Die Kontakte in dieser Auswahl müssen sich gegenseitig ausschließen. Im Gegensatz dazu schließen sich die Kontakte, die mit dem zuerst

beginnenden Kontakt überlappen, nicht zwangsmäßig aus. Im Beispiel aus Abbildung 12 gehören in diese Gruppe Kontakt 1 und Kontakt 2.

Als nächstes wird für jeden Kontakt der nachfolgende Kontakt in der Gruppe ermittelt. Im Beispiel besitzt Kontakt 2 als einziger Kontakt einen Nachfolger in seiner Kontaktgruppe. Alle anderen Kontakte besitzen keinen nachfolgenden Kontakt innerhalb ihrer Gruppe. Anschließend werden alle Kontakte aus der Auswahl nach ihrem nachfolgenden Kontakt in Kategorien eingeteilt. In Kontaktgruppe 1 aus Abbildung 12 gibt es zwei Kategorien. In einer Kategorie sind Kontakt 1 und Kontakt 3, weil beide keinen nachfolgenden Kontakt besitzen. In der anderen Kategorie befindet sich Kontakt 2, dem Kontakt 3 folgt.

Weil alle Kontakte einer Kategorie den gleichen nachfolgenden Kontakt besitzen, können sie in Bezug auf den folgenden Teil der Kontaktfolge als äquivalent angesehen werden und es kann frei zwischen ihnen gewählt werden. Aus jeder Kategorie wird der Kontakt mit der höchsten übertragenen Datenmenge gewählt. Anschließend wird beginnend mit dem nachfolgenden Kontakt der Rest der Kontaktgruppe wieder mit der Funktion `best_contact_sequence_from_sorted_group()` rekursiv auf die beste Kontaktfolge untersucht. Der beste Kontakt der Kategorie und seine beste nachfolgende Kontaktfolge ergeben zusammen die maximal übertragbare Datenmenge einer Kategorie.

Aus den Kategorien wird die Kontaktfolge ausgewählt, mit welcher die höchste Datenmenge übertragen werden kann. Die Kontaktfolge dieser Kategorie ist gleichzeitig die beste Kontaktfolge, welche mit der Gruppe an Kontakten erreicht werden kann.

Die besten Kontaktfolgen der Gruppen ergeben in chronologischer Reihenfolge die beste Kontaktfolge der ursprünglichen Menge an Kontakten.

4.5 Ausgabe

4.5.1 Ausgabewerte

Bei der Ausgabe wird für jede im Analysemodus betrachtete Entität jeweils die erreichte Datenmenge ausgegeben. Im Analysemodus „alle“ wird die mit allen Antennen maximal erreichbare Datenmenge angezeigt. Durch ein Balkendiagramm wird der Anteil an der über den Analysezeitraum angepeilten Datenübertragungsrate dargestellt. Zusätzlich lässt sich die optimale Kontaktfolge der Entität durch einen Klick auf das Menü mit den drei Punkten einblenden.

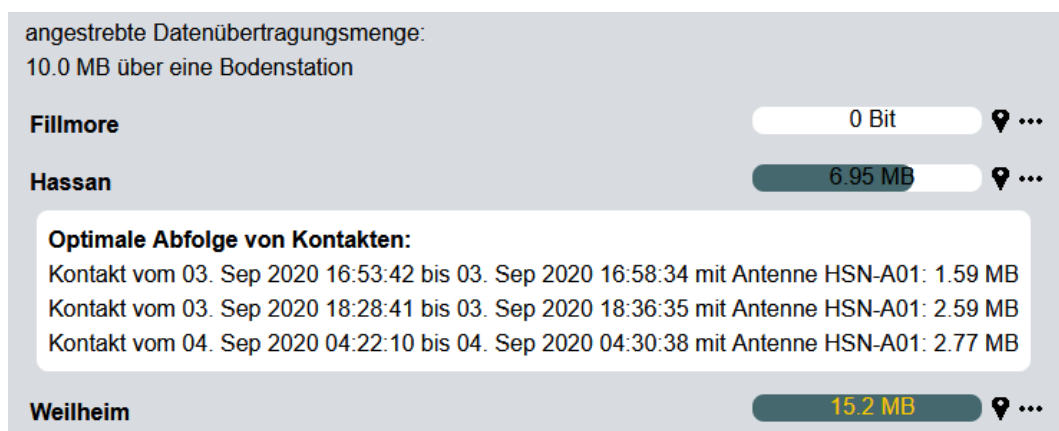


Abbildung 13 – Ausschnitt aus der Ausgabe im Analysemodus "Bodenstationen". Bei der Bodenstation "Hassan" ist die optimale Kontaktfolge eingeblendet

4.5.2 Wechsel zwischen Eingabe und Ausgabe

Für die Ein- und Ausgabe wird dasselbe Template verwendet. Alle zur Eingabe verwendeten HTML-Elemente besitzen die Klasse `input-mode`, alle zur Ausgabe verwendeten Elemente die Klasse `output-mode`. Zum Wechsel zwischen Eingabe- und Ausgabemodus werden die entsprechenden Elemente durch Setzen der Eigenschaft `style.display` auf `"none"` ausgeblendet und durch das Setzen der Eigenschaft auf `""` eingeblendet. Der aktuelle Ausgabemodus wird in der globalen JavaScript Variable `displayMode` gespeichert. Zwischen Eingabe- und Ausgabe kann durch einen Knopf in der oberen rechten Ecke gewechselt werden.

Beim ersten Aufrufen der Applikation, wo noch keine Analyse gestartet wurde, wird ausschließlich der Eingabebereich angezeigt und der Knopf

zum Wechseln zwischen den Modi ist ausgeblendet. An seiner Stelle lassen sich durch einen anderen Knopf festgelegte Daten zum Testen der Anwendung in die Eingabefelder eintragen.

Nach dem Abschicken von Daten zur Analyse wird das gleiche Template erneut aufgerufen. Dabei wird übergeben, dass ein Ergebnis vorliegt, und, dass die Anwendung im Ausgabemodus gestartet wird. Der nahtlose Wechsel zum Eingabemodus ermöglicht das Überprüfen und gegebenenfalls das Ändern der Eingabe.

4.5.3 Ausgabe in Cesium

4.5.3.1 Darstellung der Satellitenlaufbahn

Die Darstellung der Satellitenlaufbahn im Cesium-Viewer erfolgt durch ein Objekt der Klasse `Cesium.sampledPositionProperty`. Einzelne Zeitpunkte werden zusammen mit der Position des Satelliten zum entsprechenden Zeitpunkt hinzugefügt. Die Positionen zwischen den gegebenen Zeitpunkten werden von Cesium zur flüssigen Darstellung des Orbits angenähert.

```
1  <script>
    // sampledPositionProperty erstellen
    var sampledPositionProperty =
        new Cesium.SampledPositionProperty(
5      // Erde als Bezugssystem
        Cesium.ReferenceFrame.FIXED
        )
    // Positionen mit Zeitpunkt hinzufügen
    {% for position in satellite_positions %}
10     sampledPositionProperty.addSample(
        // Zeitpunkt
        Cesium.JulianDate.fromIso8601(
            "{{ position.time }}"
        ),
15     // Position
        Cesium.Cartesian3.fromDegrees(
            {{ position.lon }},
            {{ position.lat }},
            {{ position.alt }}
20     )
    {% endfor %}
    var satellite = new Cesium.Entity(...)
    satellite.position = sampledPositionProperty
</script>
```

Quelltextbeispiel 25 - Darstellen der Satellitenlaufbahn mit `Cesium.SampledPositionProperty`

Die Berechnung der Satellitenpositionen erfolgt in der Funktion `orbitscalculator.analysis.calculate_satellite_positions`. Beginnend mit dem Start des Analysezeitraumes werden die Positionen des Satelliten im inertialen Bezugssystem zu 100 Zeitpunkten innerhalb des ersten Orbits berechnet. Die Dauer eines Orbits des Satelliten kann aus dem TLE

entnommen werden. Die Zahl 100 wurde gewählt, weil sie einen guten Kompromiss zwischen Laufzeit der Berechnung und flüssiger Darstellung in Cesium darstellt. Aus Sicht eines Beobachters im inertialen Bezugssystem bleibt der Orbit des Satelliten über die weiteren Durchläufe seiner Bahn fast konstant. Die Abweichungen vom idealen Orbit werden durch die Berechnung der Kontakte mit Skyfield bei der Analyse der Datenübertragungsmengen mit beachtet. Zur visuellen Darstellung ist diese Genauigkeit nicht notwendig. Durch das Verschieben der Zeitpunkte der Positionen aus dem ersten Orbit um natürliche Vielfache der Dauer eines einzelnen Orbits, können die weiteren Positionen im Analysezeitraum ermittelt werden. Es werden so lange weitere Positionen berechnet, bis das Ende des Analysezeitraumes erreicht wurde. Beim Umrechnen der Positionen vom inertialen ins erdgebundene Bezugssystem wird schließlich die Verschiebung des Orbits aus Sicht der Bodenstation mit einbezogen. Alle ermittelten Positionen werden dem Template übergeben und in JavaScript zur `SampledPositionProperty` hinzugefügt.

4.5.3.2 Darstellung der Kontakte

Die Darstellung der Kontakte zwischen Satellit und Bodenstation erfolgt über Linien zwischen dem Satelliten und der Bodenstation zu dem Zeitpunkt, zu welchem ein Kontakt vorliegt. Dabei werden Kontakte orange gekennzeichnet, die Teile einer Kontaktfolge sind, mit welcher je nach Analysemodus die maximale Datenmenge übertragen werden kann. Kontakte, welche für die Datenübertragung nicht optimal sind, werden mit weißen Linien dargestellt. Die Kontakte sind Objekte von `Cesium.Entity`, welche das Attribut `polyline` besitzen, was festgelegt, dass sie eine Linie zwischen dem Satelliten und der Bodenstation darstellen. Um die Linien nur anzuzeigen, wenn Kontakte stattfinden, wird ein Objekt der Klasse `Cesium.TimeIntervalCollectionProperty` verwendet. Dabei werden alle Zeiträume, in denen ein Kontakt und in welchen kein Kontakt stattfindet, angegeben. Eine `TimeIntervalCollectionProperty` wird auch verwendet, um die Beschreibung des korrekten Kontaktes beim Klicken auf die Linie anzuzeigen.

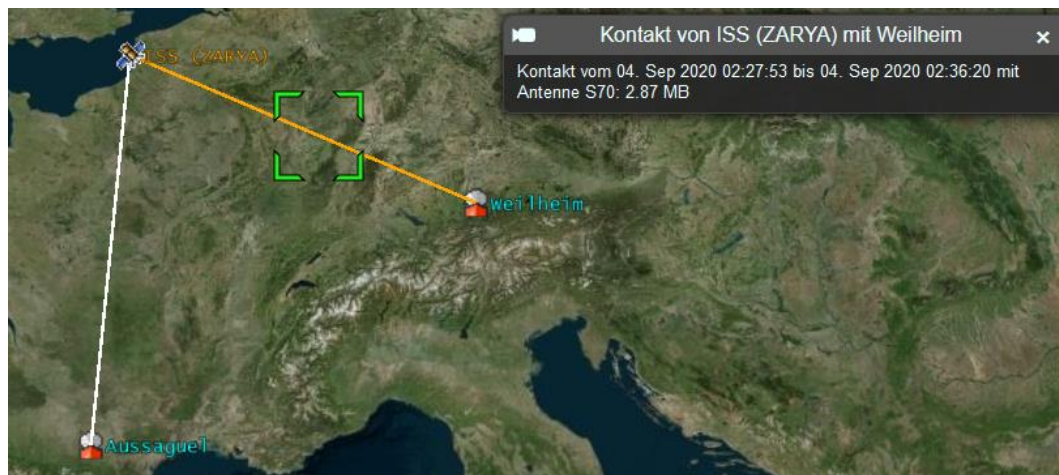


Abbildung 14 - Darstellung der Kontakte durch Linien zwischen Satellit und Bodenstation. Zwischen dem Satelliten und der Bodenstation Weilheim findet ein optimaler Kontakt statt. Die Beschreibung des Kontaktes wurde durch Klicken auf die Linie eingeblendet.

Die Bewegung des Satelliten auf dem Orbit und der Ablauf der Kontakte werden mit Cesium abgespielt. Die Kontrolle des Ablaufes erfolgt über die in Abbildung 15 dargestellte Navigationsleiste. Durch den grün hinterlegten Knopf kann die Simulation pausiert oder gestartet werden. Mit dem blauen Schieberegler auf der rechteckigen Leiste kann der dargestellte Zeitpunkt innerhalb des Analysezeitraumes ausgewählt werden. Die Ablaufgeschwindigkeit kann über das graue Dreieck am Rand des runden Anzeigeelementes geändert werden.



Abbildung 15 - Cesium Navigationsleiste

4.6 Laufzeit

Die Webanwendung soll zur Analyse der Datenübertragung bei einer Satellitenmission über lange Zeiträume genutzt werden. Dementsprechend ist auch die Laufzeit der Anwendung selbst für die Praktikabilität von großer Bedeutung. Zur Einschätzung dessen wurden die Laufzeiten einzelner Programmteile ermittelt. Dazu wurden die in den Testdaten vorgegebenen Werte verwendet und ausschließlich alle Antennen der Bodenstation Weilheim ausgewählt. Die Tests wurden auf einem durchschnittlichen Windows Desktop Computer mit dem Django Entwicklungsserver durchgeführt. Die untersuchten Zeiträume haben vom 03.09.2020 12:00 Uhr an eine Dauer von 1, 3, 6, 14, 27, 55, 90 und 365 Tagen. Die Laufzeiten wurden aus der Differenz der Uhrzeit vor und nach dem Ausführen der entsprechenden Funktion ermittelt.

```
1  from datetime import datetime
    # Startzeit
    t1 = datetime.now()
5  # zu Testende Funktion
    tested_function()
    # Endzeit
    t2 = datetime.now()
    #Laufzeit
10 runtime = t2 - t1
```

Quelltextbeispiel 26 - Ermittlung der Laufzeit einer Funktion in Python

Die in `orbitscalac.analysis.py` wurden die Funktionen `determine_best_contact_sequence()` und `Analysis.analyse()` untersucht. Die Funktion `determine_best_contact_sequence()` wird zur Ermittlung der besten Kontaktfolge aus einer Menge von Kontakten genutzt. `Analysis.analyse()` enthält die gesamte Kernfunktionalität der Analyse. Außerdem wurde die Funktion `orbitscalac.antenna.Antenna.determine_contacts()` untersucht, welche zur Ermittlung der Kontakte mit Skyfield verwendet wird.

Analyse- zeitraum in d	Laufzeit in s		
	Ermittlung der Kontakte	Ermittlung der besten Kontaktfolge	gesamte Analyse
1	0,011	0,02	0,20
3	0,024	0,02	0,43
6	0,042	0,15	0,92
13	0,075	0,36	1,79
27	0,150	0,88	3,75
55	0,290	1,82	7,77
90	0,445	3,14	12,7
365	2,215	14,2	55,0
3651	18,3	134	528

Abbildung 16 - Tabelle mit Ergebnissen der Laufzeitanalyse

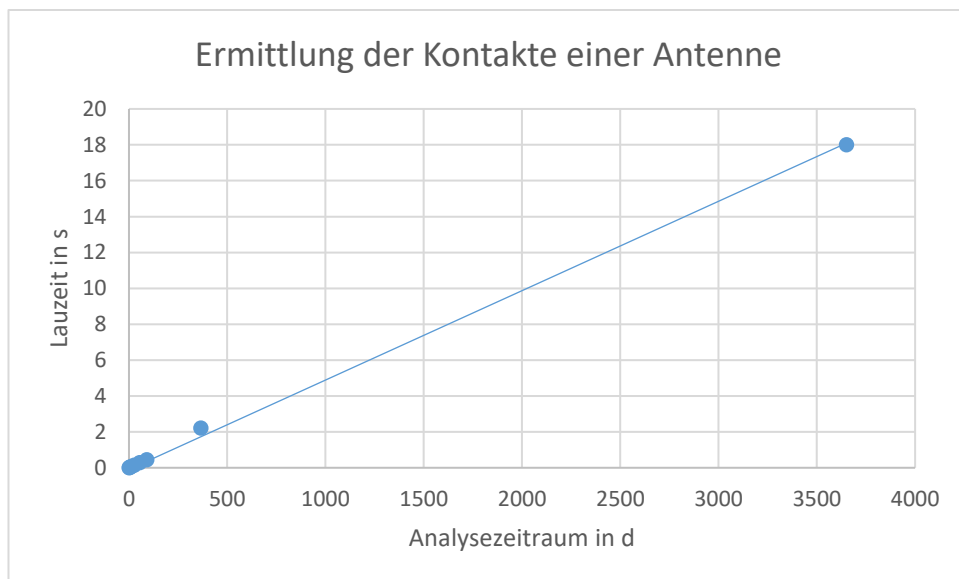


Abbildung 17 – Diagramm der Laufzeit der Funktion `orbitalcalc.antenna.Antenna.determine_contacts()` in Abhängigkeit der Länge des Analysezeitraumes

Die Abhängigkeit der Laufzeit vom Analysezeitraum ist bei der Ermittlung der Kontakt einer Antenne ungefähr linear.

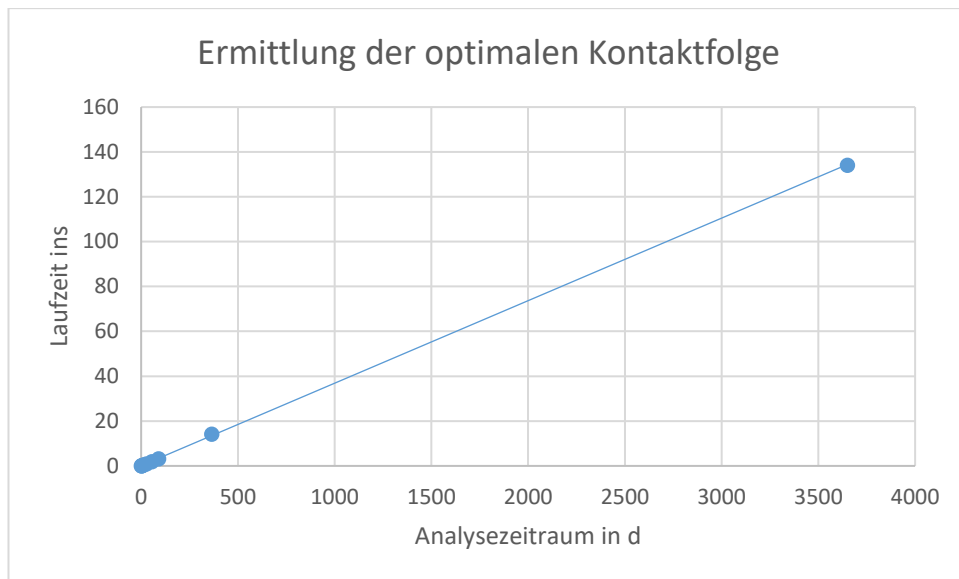


Abbildung 18 - Diagramm der Laufzeit der Funktion `orbitscal.analysis.determine_best_contact_sequence()` in Abhängigkeit der Länge des Analysezeitraumes

Die Abhängigkeit der Laufzeit der Ermittlung der optimalen Kontaktfolge ist in den Tests ebenfalls ungefähr linear. Bei vielen Überlappungen zwischen den Kontakten könnte die Laufzeit allerdings stärker ansteigen.

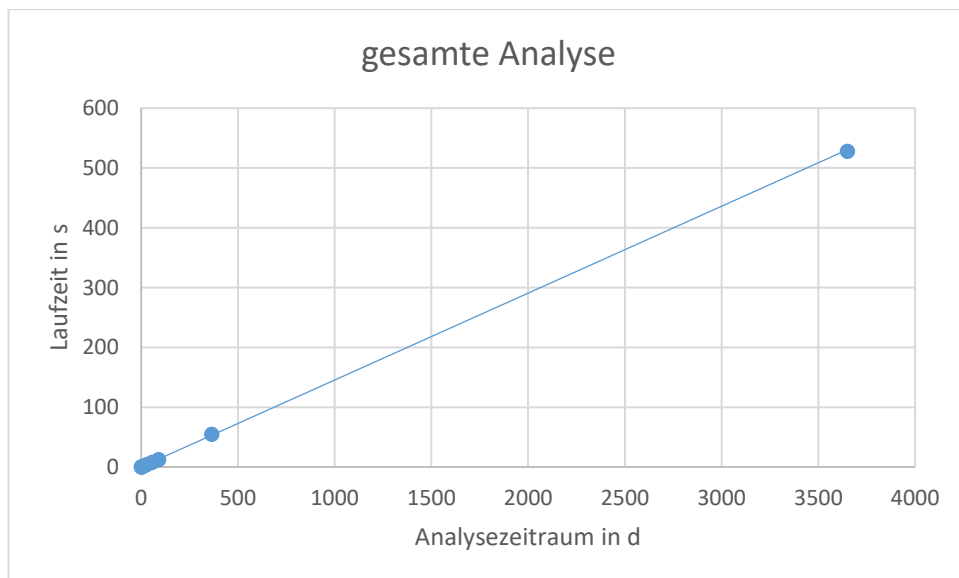


Abbildung 19 - Diagramm der Laufzeit der Funktion `orbitscal.analysis.Analysis.analyse()` in Abhängigkeit der Länge des Analysezeitraumes

Der lineare Trend spiegelt sich ebenfalls in der Abhängigkeit der gesamten Analyse von der Laufzeit wider.

Die Ermittlung der Kontakte erfolgte ursprünglich mit der Python Softwarebibliothek Pyephem. Die Positionen des Satelliten im Analysezeitraum wurden im Abstand eines bestimmten Zeitintervalls ermittelt. Für jede Position wurde dann ermittelt, ob zu diesem Zeitpunkt ein Kontakt stattfinden kann oder nicht. Angrenzende Zeitpunkte, bei denen potenziell ein Kontakt stattfinden kann, wurden dann zu einem Kontakt zusammengefügt. Der Nachteil war, dass Beginn und Ende eines Kontaktes maximal mit der Genauigkeit des gewählten Abstandes zwischen den Zeitpunkten bestimmt werden konnten. Kontakte haben im niedrigen Erdborbit eine Dauer von einigen Sekunden bis wenigen Minuten. Ein zeitlicher Abstand zwischen den einzelnen berechneten Positionen von 60s ist angesichts der geringen Dauer der Kontakte zu ungenau. Zugleich war die Laufzeit dieser Lösung der Kontaktermittlung deutlich höher als die Umsetzung mit Skyfield.

Das Ergebnis der Laufzeitanalyse ist, dass die Laufzeit der Anwendung unter den Testbedingungen auch über große Analysezeiträume nutzerfreundlich bleibt. Treiber für die benötigte Laufzeit ist die Darstellung der Positionen des Satelliten in seinem Orbit.

Bei der Analyse eines Zeitraumes von 3651 Tagen, ist ein MemoryError beim Einfügen der Liste mit den Satellitenpositionen ins Template aufgetreten, da der von der Liste der Positionen benötigte Speicherplatz zu groß war. Um dieses Problem zu umgehen und die Laufzeit insgesamt zu verbessern, wurde die Anwendung so abgeändert, dass die Satellitenpositionen ab einem Analysezeitraum von 100 Tagen nicht mehr berechnet und angezeigt werden. In einer zukünftigen Erweiterung der Anwendung bietet sich an, dem Nutzer die Option zu geben, ob Kontakte bei einer Analyse dargestellt werden sollen.

5 Fazit und Ausblick

Die als Ergebnis dieser Arbeit entstandene Webapplikation ermöglicht das Untersuchen der Datenübertragungsmengen bei wissenschaftlichen Satellitenmission. Mit Eingabe von Missions- und Satellitendaten und der Auswahl von Antennen, Bodenstationen oder Betreibern aus der erstellten Datenbank werden die erreichbaren Datenübertragungsmengen ermittelt und ausgegeben. Anhand dessen kann eine Auswahl der für die Satellitenmission genutzten Bodenstationen getroffen werden oder die Realisierbarkeit der Datenübertragung bei einem bestehenden Missionsentwurf geprüft werden.

Die in einem einfachen Test ermittelte Laufzeit der Anwendung ermöglicht eine praktikable Nutzung. Als Erweiterung sollte für lange Betrachtungszeiträume die Option gegeben werden, die grafische Darstellung auszulassen.

Die Korrektheit der Ergebnisse konnte nicht anhand von Beispielmmissionen geprüft werden, da keine Daten zum Testen der Anwendung vorlagen. Die Korrektheit und Praktikabilität der Anwendung kann daher durch den Testeinsatz im Bereich der Missionsplanung beim German Space Operations Center geprüft werden. Dafür bietet sich auch die Entwicklung einer zweisprachigen Version an, welche zusätzlich auf Englisch genutzt werden kann.

Die Anwendung kann um einen exakten Algorithmus zur Berechnung der möglichen Datenübertragungsraten erweitert werden, indem Kodierung, Modulation, Polarisierung und Dopplereffekt zusätzlich berücksichtigt werden. Eine Erweiterung um die Betrachtung der Datenübertragung im Uplink bietet sich ebenfalls an.

6 Abkürzungsverzeichnis

Abkürzung	Bedeutung
API	Programmierschnittstelle
DLR	Deutsches Zentrum für Luft- und Raumfahrttechnik
GSOC	Deutsches Raumfahrt Kontrollzentrum / German Space Operations Center
HTTP	Hypertext Transfer Protocol
TLE	Two Line Elements / Two line element set

7 Fachworterklärungen

7.1 Astronomie und Satellitenmissionen

Begriff	Erklärung
housekeeping data	vom Satelliten übertragene Daten über den Status seiner Systeme
Inertialsystem	Bezugssystem, in welchem die Trägheitsgesetze gelten
Kulmination	Höchststand eines Himmelskörpers aus Sicht des Betrachters
Link	Teil des Frequenzbereiches, welchen eine Antenne abdeckt
Propagationsmodell	physikalisches Modell zur Berechnung der Position eines Satelliten
Two Line Elements / Two line element set	Format der Bahndaten eines Satelliten

7.2 Physikalische Größen

Größe	Formelzeichen	Beschreibung
Abstand	d	Länge der Strecke zwischen zwei Punkten
Antenna gain-to-noise-temperature	G/T	Größe zur Beschreibung der Empfangsleistung einer Antenne
Bandbreite	B	Frequenzbereich, welcher zur Datenübertragung genutzt wird
Boltzmann-Konstante	k_B	$1,380649 \cdot 10^{-23} \frac{J}{K}$
Datenübertragungsrate	C	Datenmenge, welche in einer Zeitspanne übertragen wird
energy per bit to noise power spectral density ratio	E_b/N_0	Größe zur Beschreibung des Signal-Rausch-Verhältnisses pro übertragenem Bit

äquivalente isotrope Strahlungsleistung	<i>EIRP</i>	„Produkt aus der Leistung, die der Antenne zugeführt wird, und ihrem Antennengewinn in einer gegebenen Richtung, bezogen auf eine isotrope Antenne“ (ITU, 2016)
Freiraumdämpfung	<i>FSPL</i>	bei der Übertragung von Funksignalen verlorene Sendeleistung
Frequenz	<i>f</i>	Anzahl von Wiederholungen pro Zeit
Margin	-	Betrag, um welchen die Sendeleistung die Empfangsleistung übersteigt
Vakuumlichtgeschwindigkeit	<i>c</i>	maximale Ausbreitungsgeschwindigkeit elektrischer Strahlung im Vakuum

8 Literaturverzeichnis

- Django Software Foundation. (2020). *Django Documentation*. Retrieved 05. 12., 2020, from <https://www.djangoproject.com/start/overview/>
- Django Software Foundation. (2020). *Django Documentation*. Retrieved 05. 16., 2020, from <https://docs.djangoproject.com/en/3.0/ref/databases/#sqlite-notes>
- Django Software Foundation. (2020). *Django Documentation*. Retrieved 01. 01., 2021, from <https://docs.djangoproject.com/en/3.1/topics/http/views/>
- Hoots, F. R., & Roehrich, R. L. (1988, 12. 31.). *CelesTrak*. Retrieved 12. 30., 2020, from <https://www.celestrak.com/NORAD/documentation/spacetrk.pdf>
- Huber, F. (2015). 1.3 Fundamentals of Space Communication. In D. L.-u. Raumfahrtzentrum (Ed.), *Spacecraft Operations* (p. 21). Wessling, Deutschland: Springer Verlag Wien.
- ITU. (2016). Terms and definitions. In ITU (Ed.), *Radio Regulations* (p. 16). Genf.
- NASA. (2020). *NASA Space Science Data Coordinated Archive*. Retrieved 12. 5., 2020, from <https://nssdc.gsfc.nasa.gov/nmc/spacecraft/display.action?id=1957-001B>
- Ohndorf, A. (2015). 2.1 Mission Operations Preparation. In D. L.-u. Raumfahrtzentrum (Ed.), *Spacecraft Operations* (p. 36). Wessling, Deutschland: Springer Verlag Wien.
- Rhodes, B. (2020). *rhodesmill*. Retrieved 12. 30., 2020, from <https://rhodesmill.org/skyfield/>

9 Abbildungs- und Quelltextverzeichnis

9.1 Abbildungsverzeichnis

Abbildung 1 - Algorithmus zur Berechnung der maximalen Datenübertragungsrate in Pseudocode	8
Abbildung 2- Tabelle mit übernommenen Werten zur Berechnung der Datenübertragungsrate	10
Abbildung 3 - ERD der Datenbank in Chen-Notation	13
Abbildung 4 - Ergebnis des Django Template und View	20
Abbildung 5 - Eingabebereich der Satellitendaten	21
Abbildung 6 - Eingabebereich der Analysedaten	21
Abbildung 7 - Eingabebereich Antennen, Bodenstationen und Betreiber	23
Abbildung 8 - Aufbau eines TLE https://spaceflight.nasa.gov/realdata/sightings/SSapplications/Post/JavaSOP/SSOP_Help/tle_def.html Zugriff: 07.07.2020.....	27
Abbildung 9 - Ausschnitt aus der Darstellung in der Webanwendung mit Cesium. Die Bodenstation Weilheim ist ausgewählt.	28
Abbildung 10 - Anzeigen einer Bodenstation auf der Karte	31
Abbildung 11 - Aufteilung der Zeiträume in einem Kontakt	34
Abbildung 12 - Schematische Darstellung von Kontaktgruppen	36
Abbildung 13 – Ausschnitt aus der Ausgabe im Analysemodus "Bodenstationen". Bei der Bodenstation "Hassan" ist die optimale Kontaktfolge eingeblendet	38
Abbildung 14 - Darstellung der Kontakte durch Linien zwischen Satellit und Bodenstation. Zwischen dem Satelliten und der Bodenstation	

Weilheim findet ein optimaler Kontakt statt. Die Beschreibung des Kontaktes wurde durch Klicken auf die Linie eingeblendet.....	42
Abbildung 15 - Cesium Navigationsleiste.....	42
Abbildung 16 - Tabelle mit Ergebnissen der Laufzeitanalyse	44
Abbildung 17 – Diagramm der Laufzeit der Funktion orbitscalc.antenna.Antenna.determine_contacts() in Abhängigkeit der Länge des Analysezeitraumes.....	44
Abbildung 18 - Diagramm der Laufzeit der Funktion orbitscalc.analysis.determine_best_contact_sequence() in Abhängigkeit der Länge des Analysezeitraumes	45
Abbildung 19 - Diagramm der Laufzeit der Funktion orbitscalc.analysis.Analysis.analyse() in Abhängigkeit der Länge des Analysezeitraumes.....	45

9.2 Quelltextverzeichnis

Quelltextbeispiel 1 – Django Model Klasse Link	14
Quelltextbeispiel 2 – Django Model mit Attribut vom Typ DecimalField ...	14
Quelltextbeispiel 3 – abgeleitetes Attribut in Django Model	15
Quelltextbeispiel 4 – Fremdschlüssel im Django Model	15
Quelltextbeispiel 5 - m:n Beziehung in Django.....	15
Quelltextbeispiel 6 - Datenbankabfragen	16
Quelltextbeispiel 7 – Funktion zur Prüfung, ob alle Attribute eines Eintrags gültig sind.....	17
Quelltextbeispiel 8 - Rückgabe der vollständigen Datensätze der zu einer Antenne gehörigen Links	17
Quelltextbeispiel 9 - Django View	19
Quelltextbeispiel 10 - Django Template	19
Quelltextbeispiel 11 – Django URL Mapping	20
Quelltextbeispiel 12 - Django Form Grundlagen	24
Quelltextbeispiel 13 - Django Form Auswahlfeld.....	25
Quelltextbeispiel 14 - Erstellen von Django Form im View.....	25
Quelltextbeispiel 15 - Django Form im Template	26
Quelltextbeispiel 16 - Überprüfung der Gültigkeit eines Django Forms....	26
Quelltextbeispiel 17 - Regulärer Ausdruck TLE	27
Quelltextbeispiel 18 - RegexpField TLE	28

Quelltextbeispiel 19 - Cesium Grundlagen.....	29
Quelltextbeispiel 20 - Erstellen des Cesium Entity für Bodenstation.....	30
Quelltextbeispiel 21 - Erstellen der Objekte für Satellit und Antenne in Skyfield	32
Quelltextbeispiel 22 - Ermitteln der Kontakte mit Skyfield.....	32
Quelltextbeispiel 23 - Enumeration für die Skyfield Events.....	33
Quelltextbeispiel 24 - Berechnung der relativen Position mit Skyfield.....	33
Quelltextbeispiel 25 - Darstellen der Satellitenlaufbahn mit Cesium.SampledPositionProperty	40
Quelltextbeispiel 26 - Ermittlung der Laufzeit einer Funktion in Python ...	43

Danksagung

Ich habe beim Erstellen dieser Besonderen Lernleistung viel über das analytische Arbeiten, das Programmieren, aber auch viel über mich selbst gelernt. Die in dieser Arbeit entwickelte Anwendung war mein erstes großes Softwareprojekt und zudem das erste, welches ich in Python umgesetzt habe. Ich kann sagen, dass die Besondere Lernleistung für mich persönlich ihr Ziel erfüllt hat und mit einem wesentlichen Impuls für meine zukünftige akademische und berufliche Ausrichtung gegeben hat.

Ich bedanke mich bei meiner Familie, die mich auch in schwierigen Phasen zur Arbeit an meiner Besonderen Lernleistung motiviert hat, auch wenn „ich muss an meiner BeLL arbeiten“ manchmal schon wie eine Ausrede geklungen hat.

Ebenfalls möchte ich meinem Betreuer aus der Wilhelm-Ostwald-Schule Herrn Rai-Ming Knospe für die Unterstützung in formellen und fachlichen Aspekten herzlich danken. Im Informatikunterricht bei Herrn Knospe habe ich alle fachlichen Grundlagen, die ich zur Erstellung dieser Besonderen Lernleistung brauchte, gelernt. Damit hat er maßgeblich zu meinem Interesse für die Informatik und nicht zuletzt zu dieser Arbeit beigetragen.

Mein besonderer Dank gebührt meiner externen Betreuerin Frau Diana Peters vom Institut für Datenwissenschaften des Deutschen Zentrums für Luft- und Raumfahrttechnik. Sie stand mir bei Fragen immer sofort zur Seite und hat mir viele hilfreiche Tipps gegeben. Ich hatte außerdem im Rahmen eines Praktikums die Gelegenheit, mich mit Marcin Gnat und Robert Philipp aus dem Bereich der Missionsplanung beim GSOC auszutauschen, bei denen ich mich an dieser Stelle ebenfalls herzlich bedanken möchte.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe erstellt habe.

Ich versichere, dass alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht wurden.

Außerdem erlaube ich die schulinterne Verwendung der Arbeit sowie die Verwendung der Arbeit durch das Deutsche Zentrum für Luft- und Raumfahrt.